UT2 QB

[What are the key features of Node.js? - GeeksforGeeks](https://www.geeksforgeeks.org/what-are-the-key-features-of-node-js/#:~:text=NodeJs%20facilitates%20the%20integration%20of,scripting%20and%20client%2Dside%20programming).

* What do you mean by Nodejs? What are its features?

WAP to create an app in Nodejs.

Node.js is a cross-platform runtime environment that allows you to create server-side and networking applications. Node.js apps are written in JavaScript and run on OS X, Microsoft Windows, and Linux utilizing the Node.js runtime.

Key Features of NodeJs

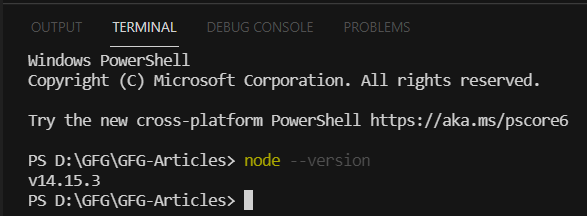
1. Asynchronous and Event-Driven: The Node.js library’s APIs are all asynchronous (non-blocking) in nature.
2. Single-Threaded: Node.js employs a single-threaded architecture with event looping, making it very scalable.
3. Scalable:
4. Quick execution of code: V8 JavaScript Runtime
5. Cross-platform compatibility: NodeJS may be used on a variety of systems, including Windows, Unix, Linux, Mac OS X, and mobile devices.
6. Uses JavaScript: Most of the engineers are already familiar with JavaScript. As a result, a designer who is familiar with JavaScript will find that working with Node.js is much easier.
7. Fast data streaming: It processes and uploads a file simultaneously, thereby saving a lot of time. As a result, NodeJs improves the overall speed of data and video streaming.
8. No Buffering: In a Node.js application, data is never buffered.

Working with Nodejs

Step 1:

Verify if node.js is installed.

node --version



Step 2:

Create package.json by using the following command to store the metadata of the project.

npm init -y

Step 3:

Now install Express in the root directory using the following command in the terminal.

npm install express --save

Step 4:

Create an app.js file in the root directory.

Contents of the app.js:

// app.js

const express = require('express'); // Importing express module

const app = express(); // Creating an express object

const port = 5000; // Setting an port for this application

// Handing the route to the server

app.get('/', function (req, res) {

res.send('Welcome to Geeksforgeeks Article');

});

// Starting server using listen function

app.listen(port, function (err) {

if (err) {

console.log("Error!!!");

}

else {

console.log("Server is running at port " + port);

}

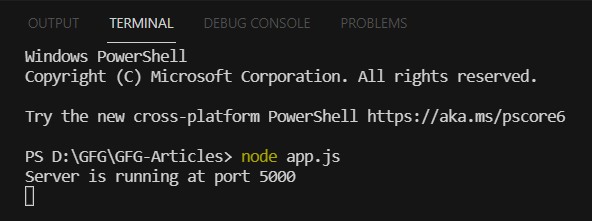
});

Step 5:

Run the above code and start the server using the following command.

>>node app.js

Output:



**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

* Explain REPL

[Node.js REPL (READ, EVAL, PRINT, LOOP) - GeeksforGeeks](https://www.geeksforgeeks.org/node-js-repl-read-eval-print-loop/)

REPL (READ, EVAL, PRINT, LOOP) is a computer environment similar to Shell (Unix/Linux) and command prompt. Node comes with the REPL environment when it installs. The system interacts with the user through outputs of commands/expressions used.

Uses: Writing and debugging codes.

The working of REPL can be understood from its full form:

Read: It reads the inputs from users and parses them into JavaScript data structure. It is then stored in memory.

Eval: The parsed JavaScript data structure is evaluated for the results.

Print: The result prints after the evaluation.

Loop: Loops the input command. To come out of NODE REPL, press ctrl+c twice

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

* How is Asynchronous Programming done in Node js?

[How to write asynchronous functions for Node.js? - GeeksforGeeks](https://www.geeksforgeeks.org/how-to-write-asynchronous-function-for-node-js/)

The asynchronous function can be written in Node.js using ‘async’ preceding the function name. The asynchronous function returns an implicit Promise as a result.

The async function helps to write promise-based code asynchronously via the event loop. Async functions will always return a value.

Await function can be used inside the asynchronous function to wait for the promise.

This forces the code to wait until the promise returns a result.

Install async from npm in Node.js using the following command:

npm i async

Use async inside your Node.js project using require() method.

Example 1:

Create an asynchronous function to calculate the square of a number inside Node.js.

* Create a project folder.
* Use the following command to initialize the package.json file inside the project folder.

npm init -y

* Install async using the following command:

npm i async

* Create a server.js file & write the following code inside it.
* Run the code using npm start

var async = require("async");

function square(x) {

return new Promise((resolve) => {

setTimeout(() => {

resolve(Math.pow(x, 2));

}, 2000);

});

}

async function output(x) {

const ans = square(x);

console.log(ans);

}

output(10);

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

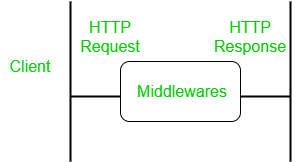
* What is Middleware in Express?

State its Types with appropriate examples

[Middleware in Express.js - GeeksforGeeks](https://www.geeksforgeeks.org/middleware-in-express-js/)

Types - [A Complete Guide on How to Build Middleware For Node.js.](https://www.turing.com/kb/building-middleware-for-node-js)

Express.js is a routing and Middleware framework for handling the different routing of the webpage and it works between the request and response cycle. Middleware gets executed after the server receives the request and before the controller actions send the response. Middleware has the access to the request object, responses object, and next, it can process the request before the server send a response. An Express-based application is a series of middleware function calls.



Types of middleware in Node.js

1. Application-level middleware

Ex. app.use

2. Router-level middleware

Ex. router.use

3. Built-in middleware

Ex. express. static, express. JSON, express.urlencoded

4. Error handling middleware

Ex. app.use(err,req,res,next)

5. Third-party middleware

Ex. bodyparser,cookieparser

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

* Explain Cookies in Express

Cookies are simple, small files/data that are sent to the client with a server

request and stored on the client side. Every time the user loads the website back, this cookie is sent with the request. This helps us keep track of the user’s actions.

The following are the numerous uses of HTTP Cookies −

Session management

Personalization(Recommendation systems)

User tracking

To use cookies with Express, we need the cookie-parser middleware. To

install it, use the following code −

npm install --save cookie-parser

Now to use cookies with Express, we will require the cookie-parser package.

cookie-parser looks at the headers in between the client and the server transactions, reads these headers, parses out the cookies being sent, and saves them in a browser. In other words, cookie-parser will help us create and manage cookies depending on the request a user

makes to the server.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

* What is Express Generator?

Express Generator is a Node.js Framework like ExpressJS which is used to create express Applications easily and quickly. It acts as a tool for generating express applications.

Features of Express-Generator:

1. It generates express Applications in one go using only one command.

2. The generated site has a modular structure that we can modify according to our needs for our web application.

3. The generated file structure is easy to understand.

4. We can also configure options while creating our site like which type of view we want to use

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

* Explain Props. What do you mean by Default props?

It is an object which stores the value of attributes of a tag to create reusable custom components. Props are arguments passed into React components.

props stand for properties.

data with props are being passed in a unidirectional flow. (one way from parent to child)

The defaultProps is a React component property that allows you to set default values for the props argument. If the prop property is passed, it will be changed. The defaultProps can be defined as a property on the component class itself, to set the default props for the class.

import PropTypes from 'prop-types';

function Car(props) {

return <h2>I am a { props.brand }</h2>;

}

Car.propTypes = {

title: PropTypes.string.isRequired

};

Car.defaultProps = {

title: 'Default'

};

}

function Garage() {

return (

<>

<h1>Who lives in my garage?</h1>

<Car brand="Ford" />

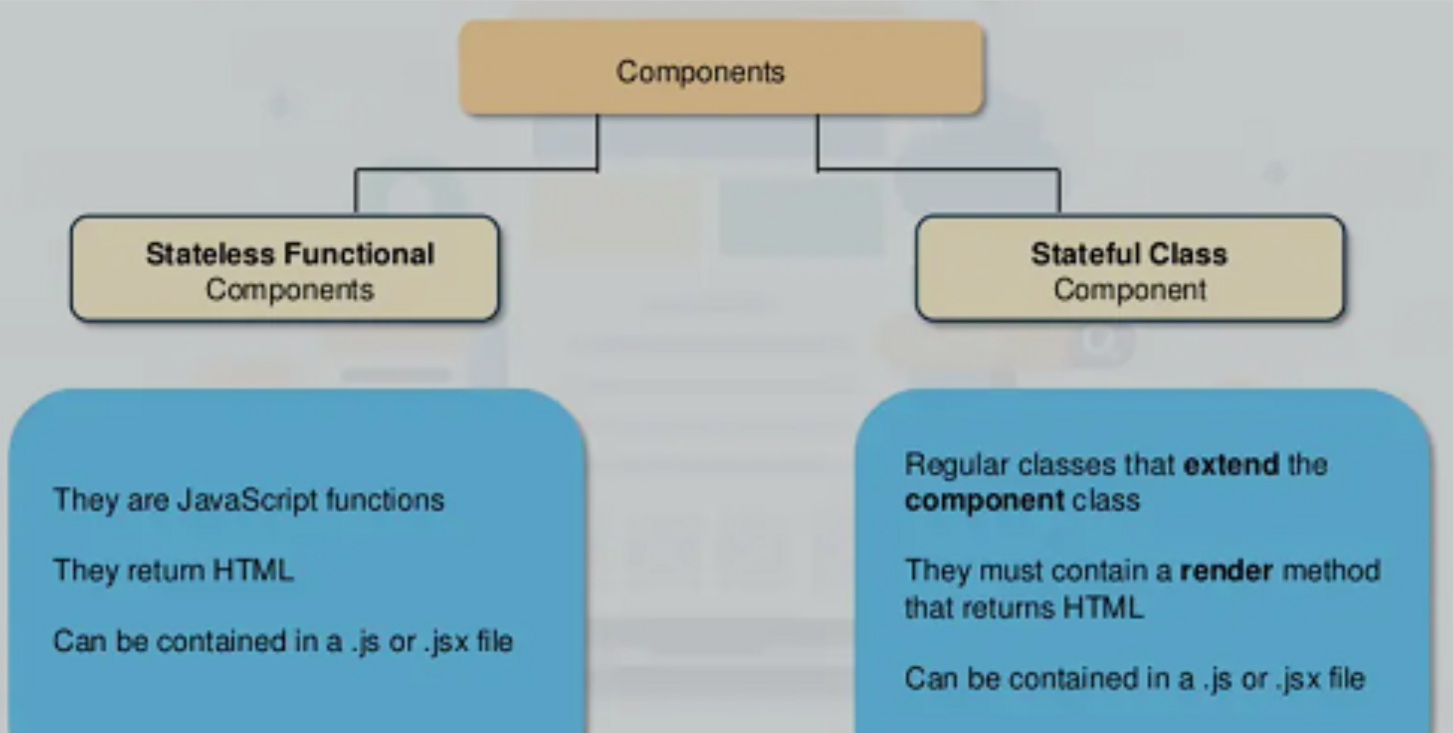
</>

);

}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

* Difference between Stateful and Stateless Component

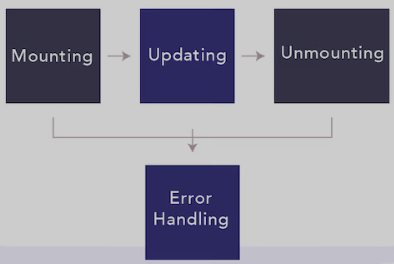


**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

* Explain the Component Lifecycle with a neat Diagram

Components are the building blocks of React application that represents a part of the user interface. Reusable

The React component goes through the following phases



Mounting Phase

This phase refers to the component’s creation. This is where the component is added to the DOM.

Updating: Updating is the stage when the state of a component is updated and the application is repainted.

Unmounting: As the name suggests. Unmounting is the final step of the component lifecycle where the component is removed from the page.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

* Explain Forms and their types with examples

Forms are an integral part of any modern web application. It allows the users to interact with the application as well as gather information from the users. Forms can perform many tasks that depend on the nature of your business requirements and logic. React offers a stateful, reactive approach to building a form. The component rather than the DOM usually handles the React form.

There are two types of form input in React.

· Uncontrolled components-

It is similar to the traditional HTML form inputs. Here, the form data is handled by the DOM itself. It maintains its own state and will be updated when the input value changes. To write an uncontrolled component, there is no need to write an event handler for every state update, and you can use a ref to access the value of the form from the DOM.

Controlled components-

A controlled component is bound to a value, and its changes will be handled in code by using event-based callbacks. Here, the input form element is handled by the react itself rather than the DOM. In this, the mutable state is kept in the state property and will be updated only with the setState() method.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**